

1/5/1

DIALOG(R) File 347:JAPIO

(c) 2003 JPO & JAPIO. All rts. reserv.

04459203 **Image available**
EMULATOR

PUB. NO.: 06 -103103 [JP 6103103 A]
PUBLISHED: April 15, 1994 (19940415)
INVENTOR(s): SUGIYAMA SHIGEKI
APPLICANT(s): FUJIKURA LTD [000518] (A Japanese Company or Corporation), JP
 (Japan)
APPL. NO.: 04-248297 [JP 92248297]
FILED: September 17, 1992 (19920917)
INTL CLASS: [5] G06F-011/26
JAPIO CLASS: 45.1 (INFORMATION PROCESSING -- Arithmetic Sequence Units)
JAPIO KEYWORD: R011 (LIQUID CRYSTALS); R131 (INFORMATION PROCESSING --
 Microcomputers & Microprocessors)
JOURNAL: Section: P, Section No. 1770, Vol. 18, No. 376, Pg. 140, July
 14, 1994 (19940714)

ABSTRACT

PURPOSE: To imitate the power source abnormal state of one computer by the other computer by imitating a function corresponding to the other computer side by an imulating means constituting of each function routine, defined by the same argument as a program for one computer.

CONSTITUTION: An emulator EM is constituted of a program group redefined so that various functions defined so as to operate in the hardware environment of a handy terminal operate in the hardware environment of a developing machine, prescribes the external specifications of each function by the same argument as a terminal exclusive function library, and also, each function inside redefines so as to emulate an operation of the handy on the developing machine. That is, the terminal exclusive function library is replaced with the emulator EM, debug DG2 of an application program is executed on the developing machine, down-load of an execution file Ef and actual machine debug on the handy terminal are omitted, and a low battery state is simulated.

Best Available Copy

(19)日本国特許庁 (J P)

(12) 公 開 特 許 公 報 (A)

(11)特許出願公開番号

特開平6-103103

(43)公開日 平成 6 年(1994) 4 月15日

(51)Int.Cl.⁵

G 0 6 F 11/26

識別記号

庁内整理番号

8323-5B

F I

技術表示箇所

審査請求 未請求 請求項の数 1 (全 11 頁)

(21)出願番号

特願平4-248297

(22)出願日

平成 4 年(1992) 9 月17日

(71)出願人 000005186

株式会社フジクラ

東京都江東区木場 1 丁目 5 番 1 号

(72)発明者 杉山 茂樹

千葉県佐倉市六崎1440番地 藤倉電線株式

会社佐倉工場内

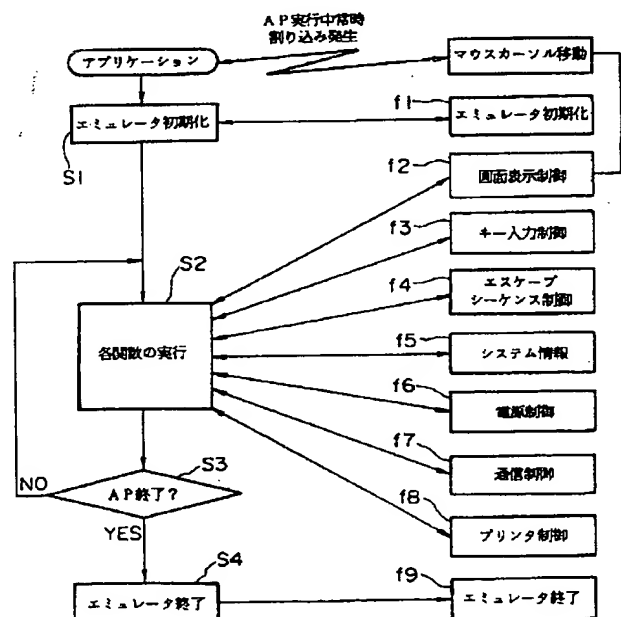
(74)代理人 弁理士 志賀 正武

(54)【発明の名称】 エミュレータ

(57)【要約】 (修正有)

【目的】 ハンディターミナル 1 のローバッテリー状態を開発マシン上でシミュレートすることができるエミュレータを実現する。

【構成】 ハンディターミナル 1 用に作成したプログラムと同一の引数で定義された各関数ルーチンから構成されるエミュレータ EM が、それぞれ開発マシン側のハードウェア構成に対応した各機能をエミュレートすると共に、このエミュレートされるハンディターミナル 1 のローバッテリー状態を、開発マシンにおける所定のキー操作によって模倣する。



【特許請求の範囲】

【請求項1】 互いにハードウェア構成が異なるコンピュータの内、いずれか一方のコンピュータ用に作成したプログラムを、他方のコンピュータ上で動作可能にするエミュレータにおいて、

前記プログラムがコールする複数のルーチンから形成されるライブラリであって、前記プログラムと同一の引数で定義された各関数ルーチンが、それぞれ前記他方のコンピュータ側のハードウェア構成に対応した各機能を模倣する模倣手段と、

所定のキー操作を検出する検出手段とを具備し、前記模倣手段は、前記検出手段によって所定のキー操作がなされた場合に前記一方のコンピュータにおける電源異常状態を、前記他方のコンピュータで模倣することを特徴とするエミュレータ。

【発明の詳細な説明】

【0001】

【産業上の利用分野】この発明は、例えば、「ハンディターミナル」と呼ばれる携帯可能なコンピュータ用のアプリケーションプログラム開発に用いて好適なエミュレータに関する。

【0002】

【従来の技術】近年、「ハンディターミナル」と呼ばれる携帯可能なコンピュータが各種実用化されている。この種のコンピュータは、ポータブルに構成されており、携行先で種々のデータ処理が可能になるため、様々な分野で使用されている。図7は、このようなハンディターミナル1の一構成例を示すブロック図である。

【0003】この図において、1aはハンディターミナル1の各部を制御するCPUである。1bは、CPU1aの基本的な動作を制御するオペレーティングシステムプログラム（以下、これをOSと略す）が記憶されるROMである。1cは対応業務用のアプリケーションプログラムがロードされると共に、該プログラムのワークエリアとして各種レジスタ値が一時記憶されるRAMである。1dはLCD（液晶表示素子）等から構成される表示回路であり、内部バスを介してCPU1aから供給される各種データを表示する。1eはLCD（液晶表示素子）上に設けられた透明タッチパネル、あるいは本体パネルに配置されるテンキーやファンクションキー等から構成される操作子であり、それぞれ各操作に応じた操作子信号を発生する。

【0004】1fは本体から着脱自在に構成されるメモリカードである。このメモリカード1fには、上述したアプリケーションプログラム、あるいは当該プログラムによって参照される各種データが記憶される。なお、メモリカード1fに記憶されるデータは、図示されていない上位コンピュータからダウンロードされるものであり、電池によってバックアップされる。1gは、例えば、モデム等から構成され、シリアルデータ通信を制御

する通信制御回路である。このような構成によるハンディターミナル1は、周知のコンピュータと同様に動作する。つまり、電源投入後にOSが立上がると共に、このOS上で対応業務用のアプリケーションプログラムがデータ処理を実行する。

【0005】ところで、ハンディターミナル1で動作するアプリケーションプログラムは、「C言語」と呼ばれる構造化プログラミング言語で記述される場合が多い。図8は、「C言語」によるプログラム開発手順を示す図である。この図に示すように、プログラム開発は、コーディング、コンパイルおよびリンクの各作業からなり、実機デバッグを経て当該プログラムの動作が検証される。なお、このコーディング、コンパイルおよびリンクの各作業は、通常、開発マシン（例えば、パーソナルコンピュータ）上で行われる。

【0006】コーディングにおいては、予め定められたシステム仕様に基づき、ソースプログラムをC言語で記述する。このソースプログラムは、ソースファイルsfとして開発マシン上に登録される。ソースファイルsfは、コンパイラCCの入力ファイルとなる。コンパイラCCでは、C言語で記述されたソースプログラムを語彙解析、構造解析および意味解析し、この結果をリロケータブルな中間コードで記述されたオブジェクトプログラムに変換する。このオブジェクトプログラムは、開発マシン上でオブジェクトファイルofとして登録される。

【0007】リンカLKでは、オブジェクトファイルofに対し、標準関数ライブラリSLと、ターミナル専用関数ライブラリTLとを結合させ、マシン語で記述された実行ファイルEfを生成する。ここで、標準関数ライブラリSLは、C言語において定義される各種の制御関数ルーチンプログラムから構成されている。また、ターミナル専用関数ライブラリTLは、ハンディターミナル1のハードウェア環境で動作するように定義された専用プログラム群から構成されるものである。

【0008】すなわち、リンカLKでは、オブジェクトプログラムにおいてコールされる種々のルーチンプログラムや専用プログラムが、上述した各ライブラリSL、TLから引用され、これらをオブジェクトプログラムに結合させる。これにより、リロケータブルな中間コードで記述されたプログラムが実行形式に変換され、実行ファイルEfとして生成される。このようにして生成される実行形式のアプリケーションプログラムは、絶対アドレス上に展開可能な形態となる。

【0009】次に、実行ファイルEfは、例えば、開発マシンから前述したメモリカード1fに書き込まれ、該メモリカード1fを介してCPU1aにダウンロードDLされる。これにより、ハンディターミナル1上で実機デバッグDG1が施される。実機デバッグDG1においては、予め策定された検査項目に従ってアプリケーショ

3

ンプログラムの動作を検証する。この実機デバッグDG 1でバグが露見した場合には、ソースプログラムの対応箇所を修正する。そして、修正されたソースプログラムは、再びコンパイル、リンク作業を経て実行ファイル化されて実機デバッグDG 1が繰り返される。

【0010】

【発明が解決しようとする課題】さて、上述したプログラム開発手順においては、ハンディターミナル1上でバグが露見する度毎に、当該バグに対応するソースプログラム部分を修正し、これを再度実行ファイル化してハンディターミナル1にダウンロードしなければならない。このため、実機デバッグDG 1には多大な工数が費やされ、結果的に開発コスト上昇を招致するという弊害がある。

【0011】そこで、こうした弊害を解決するには、上述したコーディング、コンパイル、リンクおよびデバッグからなる一連の作業を全て開発マシン上で行い、かつ、デバッグ作業時には、特に、開発マシン上でハンディターミナル1の動作状態を全て把握できるようになることが要求される。これを換言すれば、開発マシン上でアプリケーションプログラムを実行し、ハンディターミナル1の動作をエミュレートできれば、上述した欠点が解消され、効率の良いプログラム開発が可能になる訳である。

【0012】ところが、ハンディターミナル1と開発マシンとは、ほとんどの場合、ハードウェア環境が全く異なる。動作電源を例にとっても、ハンディターミナル1では、携帯性が重視されるためにバッテリー（蓄電池）や電池等が用いられるのに対し、開発マシンでは、一般に商用電源が用いられる。ハンディターミナル1本体に

用いられる電源は主電池および副電池に分かれており（図示せず）、副電池は、主電池の交換時においてRAM 1cの記憶内容が失われないようにするためのバックアップ電源である。さらに、メモ리카ード1fにも記憶内容をバックアップするために、電池が使用されている。これら電池を長時間使用すると、電源異常の状態、いわゆる、ローバッテリー状態となり、ハンディターミナル1における異常動作や動作停止の原因となる。これを防止するために、ハンディターミナル1には、ローバッテリー状態を検出する関数が備えられ、この関数によりローバッテリー状態を検出し、例えば、その旨の表示を行うような処理がなされて、使用者に注意を促すようにしている。

【0013】しかしながら、開発マシンに用いられる商用電源は、バッテリーとは異なり、ローバッテリー状態になることはないため、開発マシン上では、ハンディターミナルのローバッテリー状態における処理を模倣することができない。このため、従来では、ハンディターミナル1にてローバッテリー状態となるのを待ち、ハンディターミナル1自身によってローバッテリー状態における処理を動

4

作確認するという方法が採られている。この方法では、所望のときにローバッテリー状態を発生させることができないので、ハンディターミナル1用のプログラム、特に、ローバッテリー状態になって動作するプログラムの開発効率は極めて悪い、という問題があった。

【0014】この発明は上述した事情に鑑みてなされたもので、ハンディターミナル1のローバッテリー状態を開発マシン上で模倣できるエミュレータを提供することを目的としている。

【0015】

【課題を解決するための手段】この発明は、互いにハードウェア構成が異なるコンピュータの内、いずれか一方のコンピュータ用に作成したプログラムを、他方のコンピュータ上で動作可能にするエミュレータにおいて、前記プログラムがコールする複数のルーチンから形成されるライブラリであって、前記プログラムと同一の引数で定義された各関数ルーチンが、それぞれ前記他方のコンピュータ側のハードウェア構成に対応した各機能を模倣する模倣手段と、所定のキー操作を検出する検出手段とを具備し、前記模倣手段は、前記検出手段によって所定のキー操作がなされた場合に前記一方のコンピュータにおける電源異常状態を、前記他方のコンピュータで模倣することを特徴としている。

【0016】

【作用】この発明によれば、互いにハードウェア構成が異なるコンピュータ間において、一方のコンピュータ用に作成されたプログラムを他方のコンピュータ上で動作する可能にする模倣手段が、検出手段による所定のキー操作によって一方のコンピュータにおける電源異常状態を、他方のコンピュータで模倣する。

【0017】

【実施例】以下、図面を参照してこの発明の実施例について説明する。図1はこの発明の一実施例を適用したプログラム開発手順の概要を示す図である。この図において、図8に示す各部と共通する部分には、同一の符号を付し、その説明を省略する。図1に示す手順が図8に示した従来例と異なる点は、ターミナル専用関数ライブラリTL（図8参照）を後述するエミュレータEMに置き換え、これにより、開発マシン（パーソナルコンピュータ）上でハンディターミナル1用に作成されたアプリケーションプログラムのデバッグDG 2を行うようにした点にある。すなわち、この実施例が意図するところは、エミュレータEMを用いたことにより、従来必要とされていた実行ファイルEfのダウンロードDLと、これに応じてなされるハンディターミナル1上の実機デバッグDG 1とを省略し、かつ、開発マシン上でハンディターミナル1のローバッテリー状態をシミュレートするようにした点にある。

【0018】次に、図1のプログラム開発手順を実現するエミュレータEMの機能概要について説明する。ま

ず、エミュレータEMは、ハンディターミナル1のハードウェア環境で動作するよう定義された各種関数を、開発マシンのハードウェア環境で動作するように定義し直したプログラム群から構成されている。

【0019】エミュレータEMの機能は、図2に示すように、ターミナル専用関数ライブラリTL（図8参照）と同一である。つまり、このエミュレータEMでは、ターミナル専用関数ライブラリTLと同様の引数で各関数の外部仕様を規定し、かつ、各関数内部は、開発マシン上でハンディターミナル1の動作をエミュレートするよう定義し直している。

【0020】図2において、f1はエミュレータ初期化機能である。このエミュレータ初期化機能f1は、ソースプログラムの先頭に「emu_start()」なる関数が記述されている場合、エミュレータEMの初期設定を行うものである。したがって、図1に示すように、オブジェクトファイルofとエミュレータEMとをリンクさせる場合には、この「emu_start()」なる関数が必要になる。なお、この関数は、ターミナル専用関数ライブラリTLとリンクする際には、何等実行に影響されないものである。したがって、エミュレータ用に作成されたソースファイルsfは、そのままハンディターミナル1用の実行ファイルに変換可能になる。

【0021】次に、f2は画面表示制御機能である。この画面表示制御機能f2では、ハンディターミナル1用に定義された各関数、例えば、表示モードの設定、カーソル移動、カーソル形状あるいは表示文字のアトリビュート設定等を行う各種の関数を、それぞれ開発マシンのハードウェア環境に対応して動作させる機能である。f3はキー入力制御機能であり、ハンディターミナル1でなされるタッチパネル入力やテンキー入力を、開発マシン上におけるマウス入力やキー入力に置き換える機能である。f4はエスケープシーケンス制御機能であり、カーソル位置指定、全画面消去あるいはスクロールアップ／ダウンなどを指定するコード出力関数を、開発マシンのハードウェア環境に対応して動作させる機能である。

【0022】f5は、メインメモリ容量、ドライバ登録情報、外字登録情報等のシステム情報を取得して出力する関数を開発マシンに対応させた機能である。f6は主電源オン／オフ制御や、ローバッテリー状態検出などの動作を開発マシン上で表示させる電源制御機能である。f7は、ハンディターミナル1用に定義されたシリアルデータ通信制御を、開発マシン上でエミュレートさせる通信制御機能である。f8はプリンタ制御機能であり、ハンディターミナル1用に定義された印字フォント、改行ピッチなどを開発マシン上で制御するものである。f9は、エミュレータ終了機能であり、ソースプログラムの最後、あるいは途中で「emu_exit(n)」なる関数が記述されている場合、エミュレータEMを終了させる機能である。

【0023】次に、上記エミュレータEMを用いたプログラム開発手順と、この手順に基づき作成されたアプリケーションプログラムのエミュレート動作概要と、該エミュレート動作におけるマウス入力処理とについてそれぞれ説明する。アプリケーションプログラムを開発する際には、まず、コーディング段階でソースプログラムの先頭に「emu_start()」なる関数を記述し、かつ、該プログラムの最後に「emu_exit(n)」なる関数を記述しておく。そして、図1に示すように、ソースファイルsfをコンパイルccにかけ、オブジェクトファイルofを生成する。

【0024】次に、このオブジェクトファイルofに対して標準関数ライブラリSLおよびエミュレータEMをリンクさせる。これにより、ソースプログラムでコールされるハンディターミナル固有の各種関数が開発マシン上で動作する形で実行ファイル化される。実行ファイル化されたアプリケーションプログラムは、開発マシンの主メモリ上にロードされてハンディターミナル1の動作をエミュレートする。

【0025】アプリケーションプログラムのエミュレート動作

上記手順により作成されたアプリケーションプログラムが起動すると、開発マシンの処理は、図3に示すステップS1に進む。ステップS1では、プログラム先頭に定義された「emu_start()」なる関数に基づき、上述したエミュレータ初期化機能f1が初期設定を行う。この初期設定とは、ハンディターミナル1の起動状態をエミュレートするものであり、例えば、バックライトのオン／オフ状態、液晶パネルのコントラスト状態あるいはスピーカ音量などを設定するものである。

【0026】こうして開発マシン上でハンディターミナル1の初期状態がエミュレートされると、例えば、図4に示すように、開発マシンのディスプレイDSPにハンディターミナル1の表示画面TDが表示される。ここで、同図(イ)は、エミュレータ初期化機能f1に基づきハンディターミナル1の初期化状態がエミュレートされた表示例である。一方、同図(ロ)は、ハンディターミナル1の表示画面TDである。同図(イ)に示すように、ディスプレイDSPは、表示エリアE1、E2、E3に3分割されており、この内の表示エリアE1には表示画面TDと同一の表示がなされる。さらに、表示エリアE2には、ハンディターミナル1のタッチパネルの分割状態が表示され、表示エリアE3には、ハンディターミナル1の動作状態が表示される。

【0027】次いで、このような初期設定がなされると、開発マシンの処理はステップS2に進む。ステップS2では、与えられたイベントに応じてアプリケーションプログラムが各関数を実行する。このステップS2においては、ハンディターミナル1のタッチパネル入力をマウス入力するようキー入力制御機能f3が動作してお

7

り、これに対応してマウスカーソルを移動させる割込処理が実行される。

【0028】したがって、図4（イ）に示すように、実際のタッチパネルをエミュレートした表示エリアE1上の所定位置をクリックしてマウス入力を行なうと、この入力イベントに対応したアプリケーションプログラムを実行し、該プログラム中で定義された関数が実行される。これにより、ハンディターミナル1の動作が開発マシン上でエミュレートされる。例えば、マウスカーソルMCが、図4に示すように、表示エリアE1上の「印刷」の表示に置かれ、マウスクリックがなされると、該「印刷」処理が該開発マシン上でエミュレートされる。

【0029】このように各種処理がエミュレートされ、アプリケーションプログラム完了の旨を表わす入力がなされると、ステップS3の判断結果が「YES」となり、該プログラムの処理がステップS4に進む。ステップS4では、プログラム最後に定義された「emu_exit(n)」なる関数に基づき、エミュレート動作を終了する。

【0030】次に、図5および図6を参照し、上記ステップS2においてなされるローバッテリー状態発生処理について説明する。なお、この動作説明においては、前述したキー入力制御機能f3が起動されてキー入力可能状態にあり、かつ、この機能f3がキーバッファにセットされた文字列を、後述するローバッテリー状態発生ルーチンに引き渡すものとする。ここで、ステップS2においてアプリケーションプログラムがキー入力状態にある場合、図5に示すローバッテリー状態発生ルーチンが、割込処理によって所定期間毎に起動され、キー入力状態をつねにチェックしている。

【0031】まず、ステップSP1では、現時点において、キーバッファにデータが有るか否かが判別される。データが存在しないときには、すなわち、なんらキー操作がなされていない場合には、判別結果が「NO」となって、このルーチンにおける以後の処理が実行されることなく、処理は直ちに終了する一方、データが存在するときには判別結果が「YES」となって、処理はステップSP2に進み、キーバッファ内のデータがスキャンされ、さらに、ステップSP3に進んで、スキャンされたデータが、ローバッテリー状態を発生すべき旨のキー操作に相当するものであるか否かが判別される。

【0032】ここで、ローバッテリー状態を発生すべき旨のキー操作とは、予め決められた特定のキー操作方法をいうものであり、この実施例では、例えば、「CTRL」+「O」、「CTRL」+「B」、「CTRL」+「L」のいずれかに相当している。これらキー操作のうち、「CTRL」+「O」は、ハンディターミナル1における主電池のローバッテリー状態をエミュレートする場合に用いられ、「CTRL」+「L」は、同副電池のローバッテリー状態をエミュレートする場合に用いられ、ま

8

た、「CTRL」+「B」は、メモリカード1fのローバッテリー状態をエミュレートする場合に用いられる。なお、「CTRL」+「O」とは、「CTRL」（コントロール）キーと「O」キーとを同時に押下する操作を意味しており、これらの操作は、ハンディターミナル1側ではサポートされないものが用いられる。

【0033】ステップSP3の判別結果が「NO」であるならば、処理はステップSP6に進み、ステップSP2においてスキャンしたデータをキーバッファに戻し、このデータを他の処理にて用いるべくこのルーチンは終了する一方、判別結果が「YES」であるならば、処理はステップSP4に進み、状態フラグLB1～LB3をキー操作に対応してそれぞれオンにする。すなわち、押下されたキーが「CTRL」+「O」であるならば、状態フラグLB1がオンとなり、「CTRL」+「L」であるならば、状態フラグLB2がオンとなり、「CTRL」+「B」であるならば、状態フラグLB3がオンとなる。

【0034】そして、図4に示す表示エリアE3の「電源状態」の状態表示が「正常」からキー操作に対応する「LB1」、「LB2」、「LB3」の内のいずれかに変更され、ローバッテリー状態をエミュレートしている旨の表示がなされて、このルーチンは終了する。このように、ローバッテリー状態発生ルーチンでは、図3におけるステップS2のキー入力可能状態においてなされた所定のキー操作に応じて、状態フラグLB1～LB3のいずれかがオンとなり、これに対応して、表示エリアE3に該状態フラグの状態が表示される。

【0035】次に、図6を参照し、上記ルーチンで設定された状態フラグLB1～LB3を用いる関数例について説明する。図6に示すルーチンはローバッテリー状態取得関数の一例であり、ステップSB1において、図5に示すルーチンによって設定された状態フラグLB1～LB3が読み出され、ステップSB2において、この読み出したフラグの内容が操作者に返される。このような関数を電源制御機能f6（図3参照）が、開発マシンに対応させて動作させることによって、ローバッテリー状態検出などの動作を開発マシン上で表示することができる。これによって、従来、直接的に動作状態を検証できなかったり、目視で確認することができなかった、ローバッテリー状態におけるハンディターミナル1の動作が開発マシン上で一目瞭然となる。

【0036】このように、エミュレータEMは、ハンディターミナル1のハードウェア環境に対応させた専用関数ライブラリTLと同一の引数を備え、かつ各専用関数の機能f1～f9を開発マシンのハードウェア環境に対応させている。このため、ソースプログラムにエミュレータEMをリンクして、生成した実行ファイルEfを開発マシンにロードすれば、該開発マシン上でハンディターミナル1の動作がエミュレートされることになる。さ

らに、このエミュレート動作では、ハンディターミナル1の動作状態がディスプレイDSPに逐一表示されるので、アプリケーションプログラムのデバッグ作業が極めて効率良く行える。

【0037】また、上記実施例によれば、エミュレータ用に開発したソースプログラムがそのままハンディターミナル用のソースファイルとなり、完全互換性を備えるので、プログラム開発がすべて同一のマシン上で行なうことが可能になる。この結果、従来必要とされる実行ファイルEfのダウンロードDLと、これに応じてなされる実機デバッグDG1とが省略することができる。さらに、開発マシンでは、通常では有り得ないローバッテリー状態を、所定の操作でいつでも、エミュレートすることが可能となる。これらによって、極めて効率の良いプログラム開発が可能となる。

【0038】

【発明の効果】以上説明したように、この発明によれば、一方のコンピュータ用に作成したプログラムと同一の引数で定義された各関数ルーチンから構成される模倣手段が、他方のコンピュータ側のハードウェア構成に対応した各機能を模倣するので、一方のコンピュータにおける電源異常状態を、他方のコンピュータで模倣することができる。

【図面の簡単な説明】

【図1】 この発明の一実施例を適用したプログラム開発手順の概要を示す図である。

【図2】 同実施例におけるエミュレータEMの機能構成を示す図である。

【図3】 同実施例により作成されたアプリケーションプログラムの概略動作を示すフローチャートである。

【図4】 同実施例における表示エリアE1、E2、E3の表示例を示す図である。

10 【図5】 同実施例におけるローバッテリー状態発生ルーチンの動作を示すフローチャートである。

【図6】 同実施例におけるローバッテリー状態取得関数の動作を示すフローチャートである。

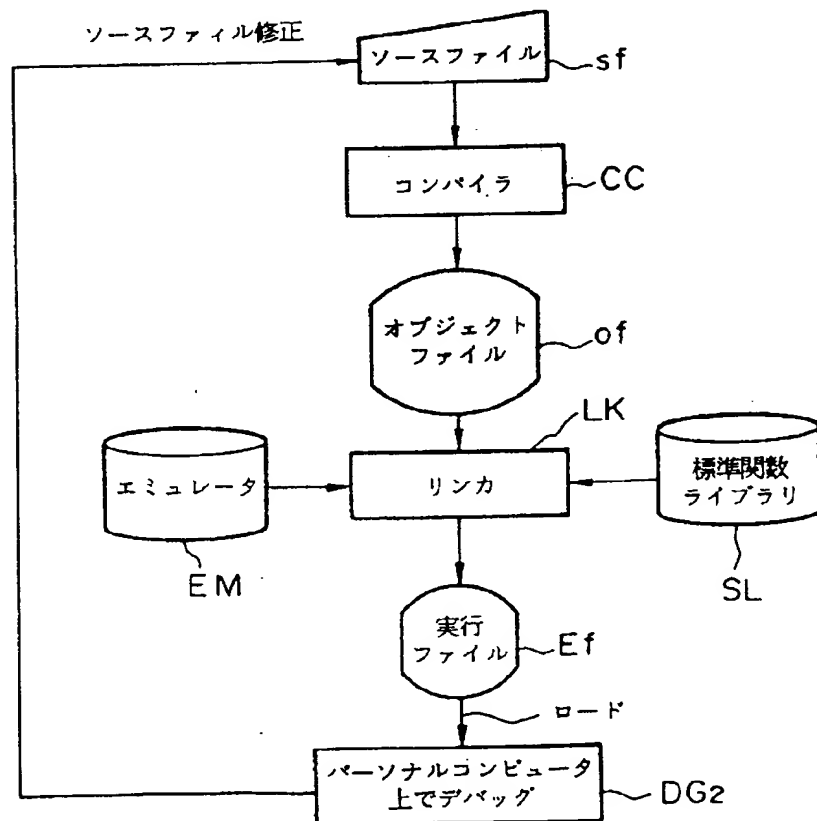
【図7】 ハンディターミナル1の一構成例を示すブロック図である。

【図8】 従来のプログラム開発手順を説明するための図である。

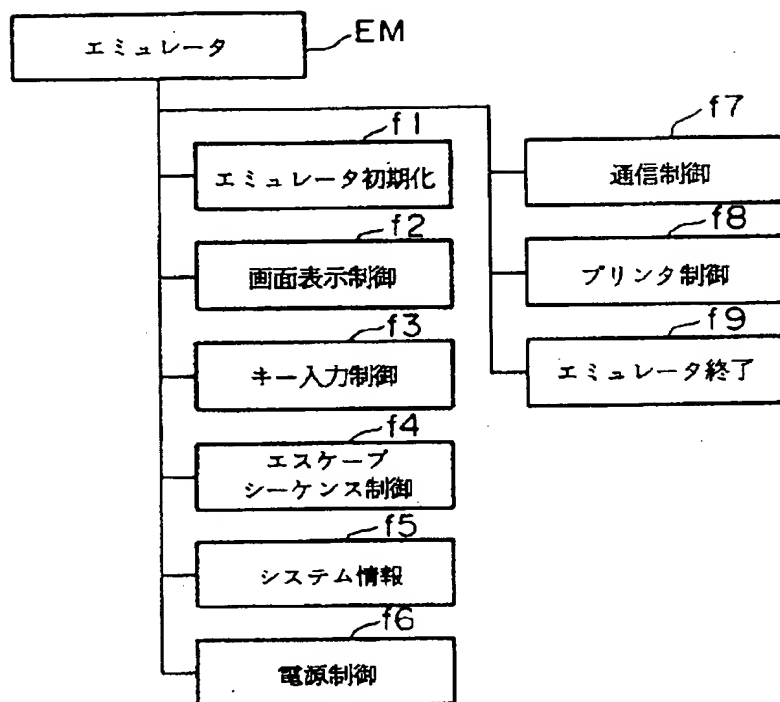
【符号の説明】

20 sf…ソースファイル、cc…コンパイル、of…オブジェクトファイル、LK…リンカ、EM…エミュレータ（模倣手段）、SL…標準関数ライブラリ、Ef…実行ファイル、f3…キー入力制御機能、f6…電源制御機能。

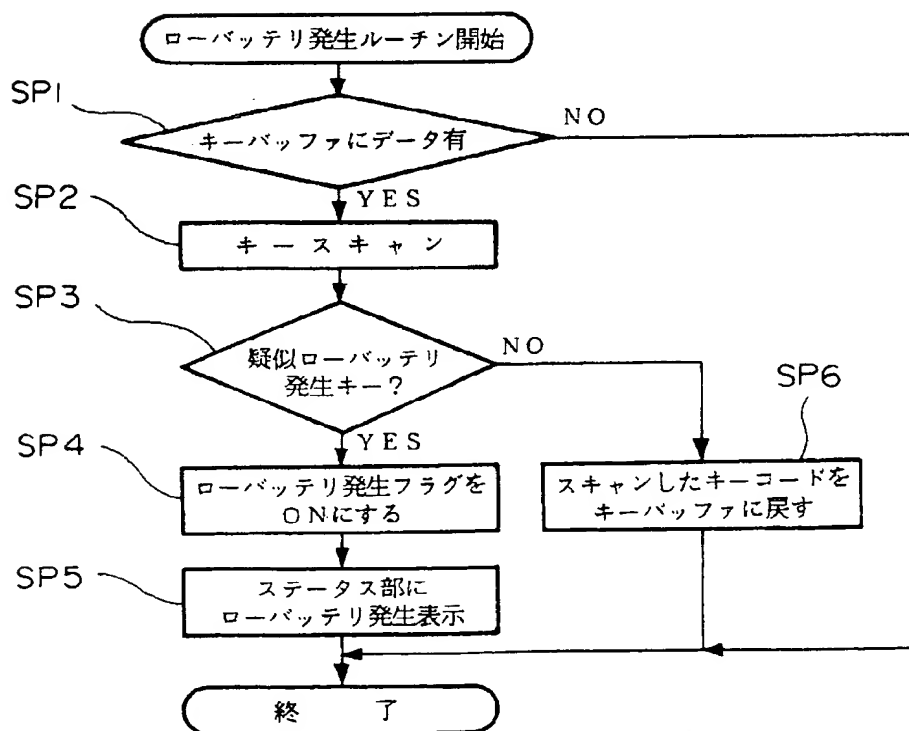
【図1】



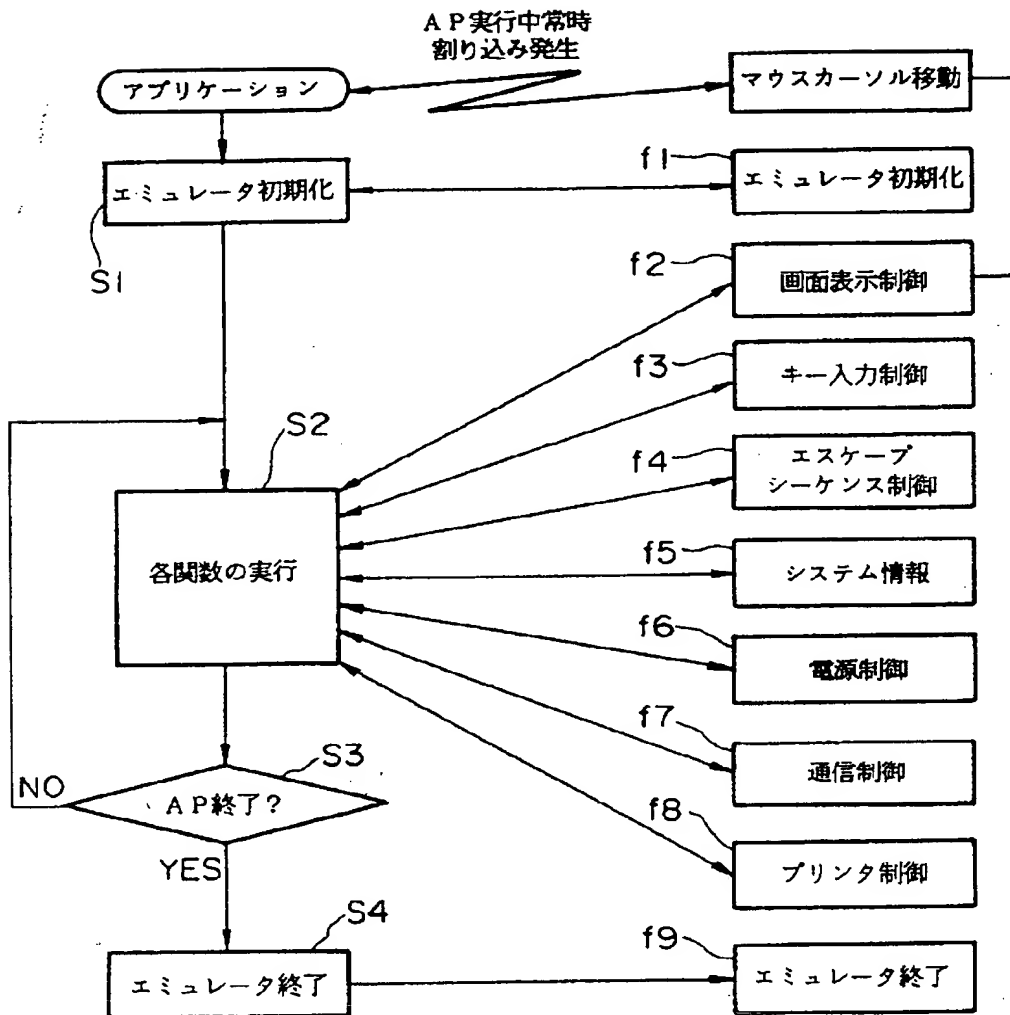
【図2】



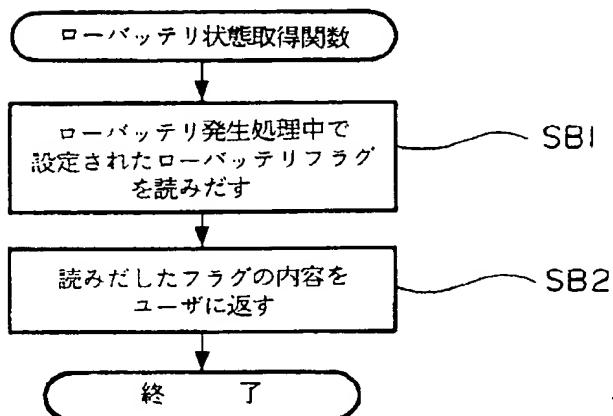
【図5】



【図3】

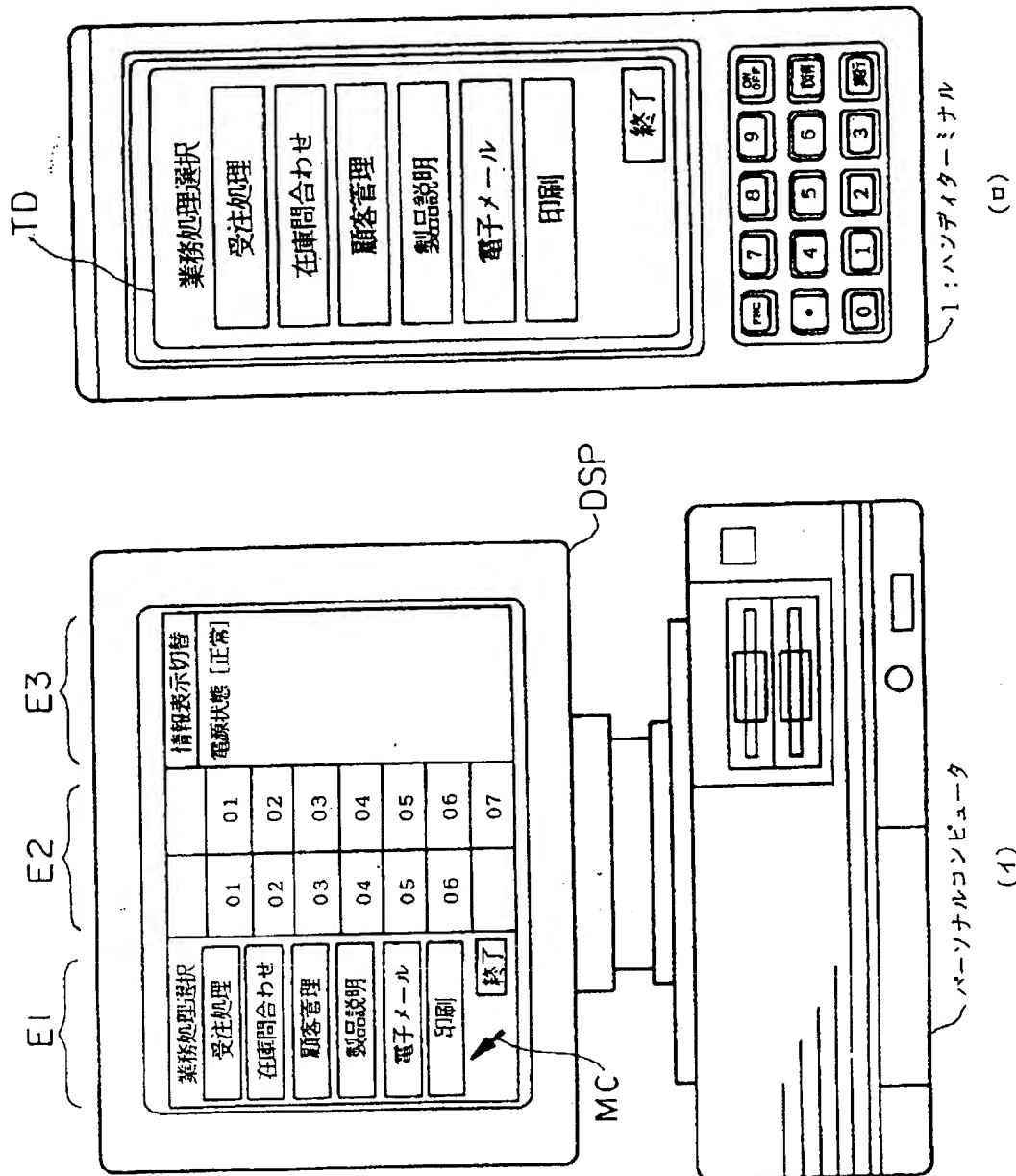


【図6】

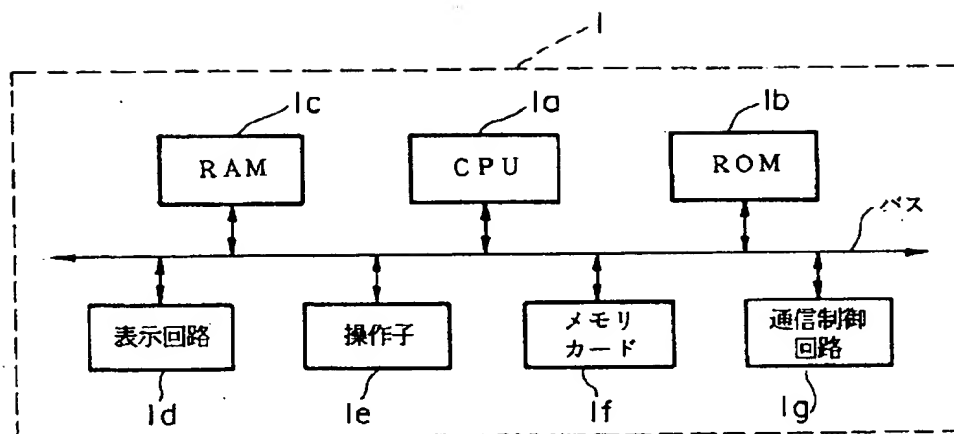


(9)

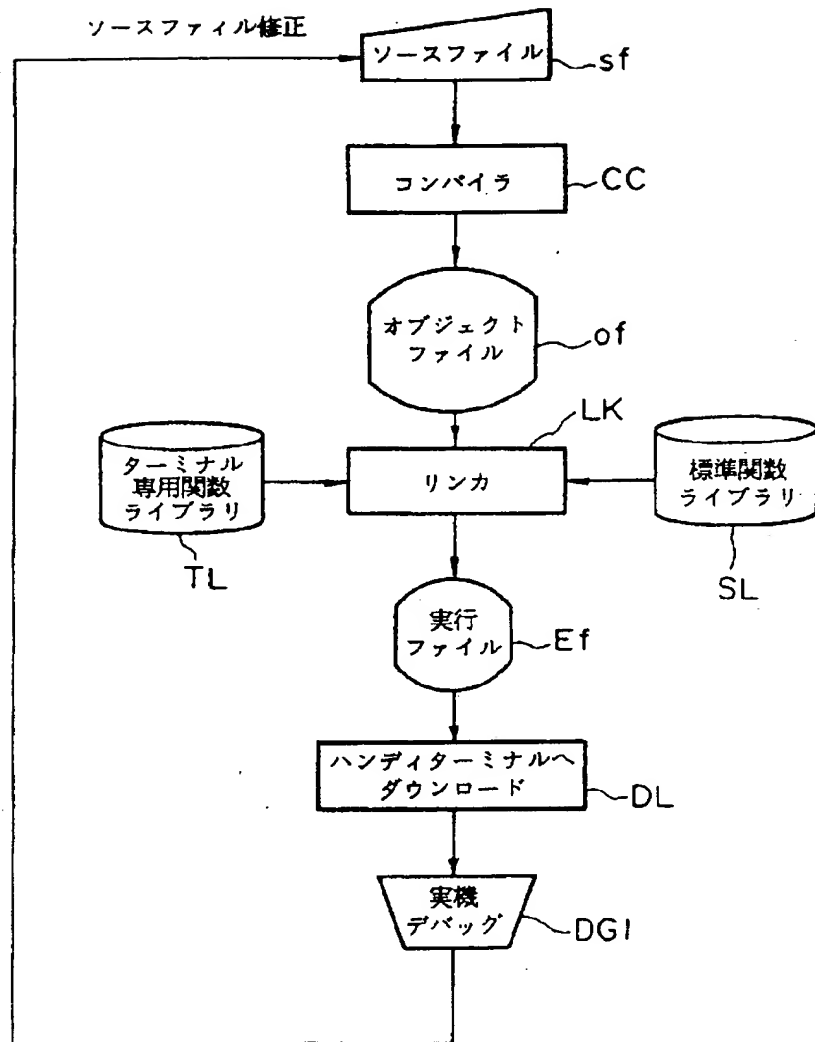
【図4】



【図7】



【図8】



**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☒ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.